Project: Bank Marketing (Campaign)

Final report

Group name: Fleifel-solo

Name: Rania Tarek Fleifel

Email: raniatarekfleifel@gmail.com

Country: Egypt

College: Cairo university Faculty of engineering

Specialization: Data science

Internship Batch: LISUM13: 30

Submission date: 20th December 2022

Github repo: Data science project-Bank Marketing Campaign

Section 1: Introduction to the problem

Problem description

Provide ABC bank with a model that enables them to target costumers who're more probable to invest in their new term deposit product.

Business understanding

As ABC launches their new term deposit product, they need their outreach teams to effectively market the product to costumers whose interactions with the bank (loans, responsiveness to offers, etc) as well as their personal standing (job stability, marital status, age, etc) show high possibility of purchasing the deposit product. The need to focus on those costumers is -at the heart of it- the bank's strategy to effectively use the marketing team's resources to spread the deposit product among interested customers.

Deposit products promise costumers a high interest rate in return to locking an amount of their money for some time. Many factors decide whether a costumer invest in such product or not. The most important is his standing in life in general. For instance, customers who have savings beyond their day-to-day spending and have sitting-money would seemingly fit the profile of a perfect costumer. If a costumer's age is 60+, he's not a very good fit since he has limited resources and is retired, hence has no income except his pension which might not offer any excess to invest in this product. Customers who are used to taking loans could benefit as well if the interest rate from their savings covers their installments to the bank. Jobs play an important role as well in defining whether a costumer is a good fit. Doctors, engineers and similar prestigious jobs that are known to pay well are good candidates, as well as individuals with a long-standing job; 20+ work in a certain company shows stability.

Section 2: Data intake and dataset choice:

Data intake

Data storage location: Bank marketing dataset (bank.zip,bank-additional.zip)

There are two datasets that could be used in this project, bank.zip (bank_full.csv, bank.csv) and bank-additional.zip (bank-additional-full.csv,bank-additional.csv). In the next section we look into both datasets and decide which to proceed with.

Tabular data details:

1) **bank-additional-full.csv**

Total number of observations	41188
Total number of files	1
Total number of features	21
Base format of the file	.csv
Size of the data	5698 KB
Unique identifier feature	No unique identifier
Dupe validation	12

2) **bank_full.csv**

Total number of observations	45211
Total number of files	1
Total number of features	17
Base format of the file	.csv
Size of the data	4502 KB
Unique identifier feature	No unique identifier
Dupe validation	No duplicates

3) **bank-additional.csv**

Total number of observations	4119
Total number of files	1
Total number of features	21
Base format of the file	.csv
Size of the data	570 KB
Unique identifier feature	No unique identifier
Dupe validation	No duplicates

4) **bank.csv**

Total number of observations	4521
Total number of files	1
Total number of features	17
Base format of the file	.csv
Size of the data	451 KB
Unique identifier feature	No unique identifier
Dupe validation	No duplicates

Dataset Choice

Will I use the older version datasets or the newer version datasets?

There are two versions of the data available. The newer version has more features/columns (21 vs. 17). However, the older version has more unique data points (45211 vs. 41176). Ideally, I would analyze the features' importance and relation to the output "y" and choose the dataset that provide more descriptive values where it matters (aka the "influential" features). However, in this project we make the decision before proceeding to indepth feature analysis.

In this subsection we look into what each version offers and conclude with the version we will proceed with.

A- Contextual understanding of features

To make an informed decision, we first begin with contextual understanding of the variables as shown in the bellow table.

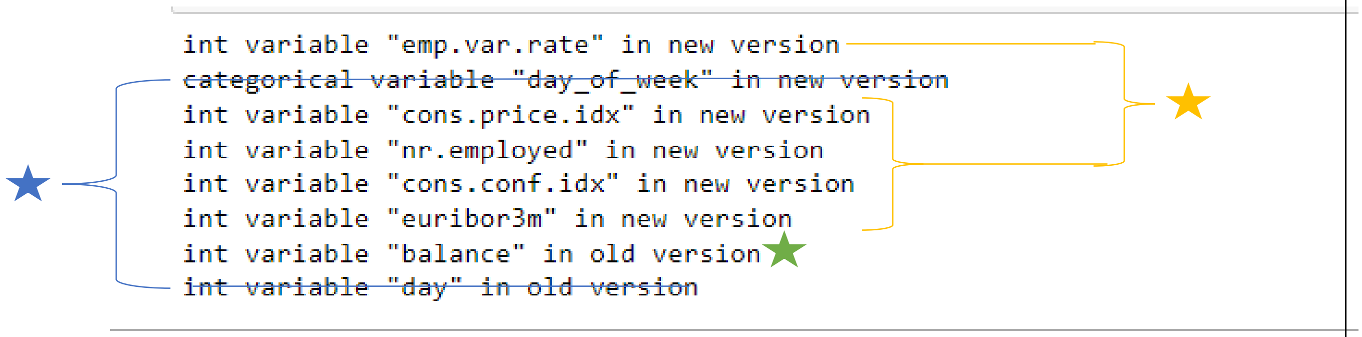
	Variable	Contextual remarks
Numeric	age	Age of costumer at most recent contact date
	balance	Average amount in euros
	*day (old dataset)	Last contact day of the month
	duration	Last contact duration in seconds
	campaign	Number of contacts with this client for this campaign
	previous	Number of contacts with this client before this campaign
	pdays	Number of days since the client was last contacted on a previous campaign *999: not previously contacted (new dataset)

	*emp.var.rate (new dataset)	Employment variation rate, with a quarterly frequency
	*cons.price.idx (new dataset)	Monthly average consumer price index
	*cons.conf.idx (new dataset)	Monthly average consumer confidence index
	*euribor3m (new dataset)	Daily three-month Euribor rate
	*nr.employed (new dataset)	Quarterly average of the total number of employed citizens
Categorical	job	Occupation of costumer
	marital	Marital status of costumer (married,divorced,single) p.s divorced=divorced/widowed
	education	Education level of costumer
	*Day_of_week (new dataset)	Last contact day of the week
	month	Last contact month
	default	If the client has credit in default?
	housing	If the client has a house loan contract (yes/no)
	loan	If the client has a personal loan contract (yes/no)
	contact	Last contact communication type
	poutcome	Outcome of previous campaign
	y	Did the client subscribe for client deposit?

B- Discrepancies in features

Next, we look into the discrepancies in the features of the two datasets and their values, we look into this for numerical and categorical features separately:

1) The discrepancies in numerical features



```
int variable "emp.var.rate" in new version
~~categorical variable "day_of_week" in new version~~
int variable "cons.price.idx" in new version
int variable "nr.employed" in new version
int variable "cons.conf.idx" in new version
int variable "euribor3m" in new version
int variable "balance" in old version
~~int variable "day" in old version~~
```

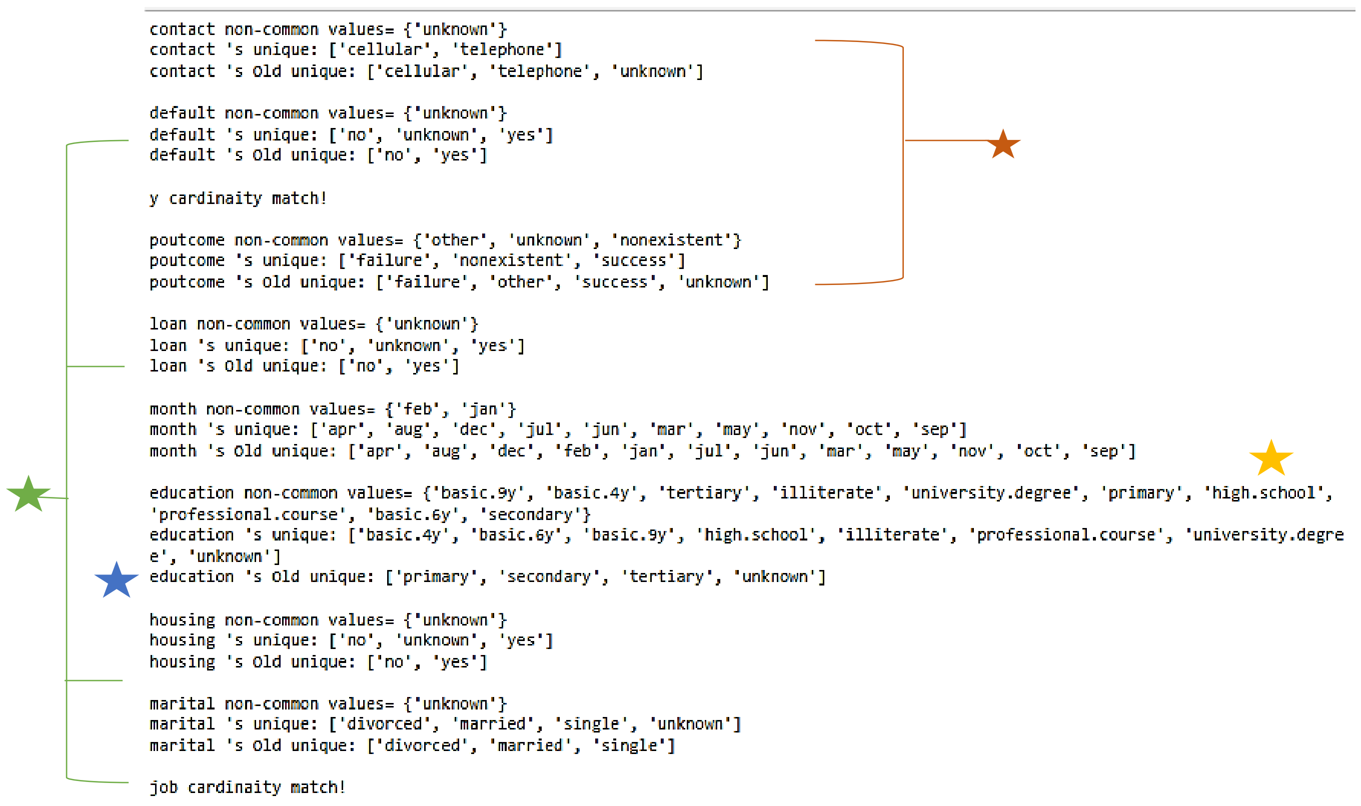
The following remarks are drawn:

★ - The variables “day” and “day_of_week” serve the same purpose

★ - The 5 variables “emp.var.rate”, “cons.price.idx”, “cons.conf.idx”, “nr.employed”, “euribor3m” are related to economic indicators

★ - The variable “balance” shows the numeric average yearly balance. It could be an indicative of how lucrative a costumer’s money is and hence how probable he would invest.

2) The cardinality in categorical features



```
contact non-common values= {'unknown'}
contact 's unique: ['cellular', 'telephone']
contact 's Old unique: ['cellular', 'telephone', 'unknown']

default non-common values= {'unknown'}
default 's unique: ['no', 'unknown', 'yes']
default 's Old unique: ['no', 'yes']

y cardinaity match!

poutcome non-common values= {'other', 'unknown', 'nonexistent'}
poutcome 's unique: ['failure', 'nonexistent', 'success']
poutcome 's Old unique: ['failure', 'other', 'success', 'unknown']

loan non-common values= {'unknown'}
loan 's unique: ['no', 'unknown', 'yes']
loan 's Old unique: ['no', 'yes']

month non-common values= {'feb', 'jan'}
month 's unique: ['apr', 'aug', 'dec', 'jul', 'jun', 'mar', 'may', 'nov', 'oct', 'sep']
month 's Old unique: ['apr', 'aug', 'dec', 'feb', 'jan', 'jul', 'jun', 'mar', 'may', 'nov', 'oct', 'sep']

education non-common values= {'basic.9y', 'basic.4y', 'tertiary', 'illiterate', 'university.degree', 'primary', 'high.school',
'professional.course', 'basic.6y', 'secondary'}
education 's unique: ['basic.4y', 'basic.6y', 'basic.9y', 'high.school', 'illiterate', 'professional.course', 'university.degre
e', 'unknown']
education 's Old unique: ['primary', 'secondary', 'tertiary', 'unknown']

housing non-common values= {'unknown'}
housing 's unique: ['no', 'unknown', 'yes']
housing 's Old unique: ['no', 'yes']

marital non-common values= {'unknown'}
marital 's unique: ['divorced', 'married', 'single', 'unknown']
marital 's Old unique: ['divorced', 'married', 'single']

job cardinaity match!
```

The following remarks are drawn:

★ - The new dataset contains more elaborate info about the education of costumers

★ - The new dataset doesn’t take months “jan” and “feb” in consideration at all.

★ - The new dataset shows the value “unknown” in a number of features “housing, marital”, “loan” and “default”

★ - The old dataset shows ambiguous values such as “unknown” and “other” in the features “poutcome” and “contact”.

Final decision: According to this short analysis, **we decide to proceed with the "Old dataset"**. This decision is in favor of the dataset that:

- Has more data points,
- Covers the whole period of analysis (all months),
- Has less ambiguous values that could later affect the predictive model
- Avoids high cardinality in some features (education).
- Has variables (day, month) which can be used to impute the features that represent economic indicators

Section 3: Data cleaning and imputation of new features

Data cleaning:

- Remove whitespaces from strings (strings=columns | column_headers)
- Handle special characters and spaces within strings
- Lower case all strings
- Ensure all columns have synchronous type of data

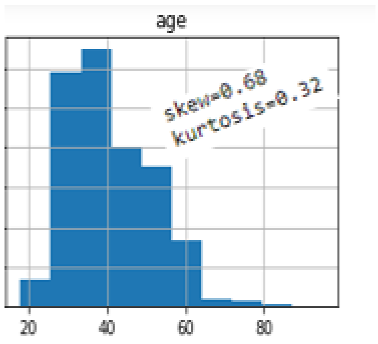
Imputation:

- Given that the data is ordered, we can deduce the year of each instance.
- Social and economic indicators from Portugal in the time span between May 2008 and Dec 2010 are quite important in our problem space; Given that these were considered global recession years. We use data available at data.nasdaq and bpstat.bportugal to represent employment, consumer price index, consumer confidence index and Euribor.

Section 4: univariate EDA

a) Numerical variables:

age

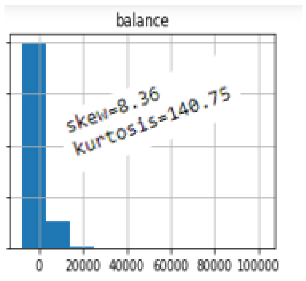


```
dtype=int64, #of nulls=0, #of zeros=0
Q1=33.0, Q2=33.0, Q3=33.0, IQR=0.0
min=18.0, max=95.0, range=range(18, 95)
mean=40.94, std=10.62, median=39.00, mad=8.74
```

- High cardinality if left as is and high dimensionality if hot-encoded
- We introduce *age_grp* feature *{0,1,2}*
  - *0*: young <25
  - *1*: adult [25,65]
  - *2*: elderly >65

** further discussed in feature engineering section

balance

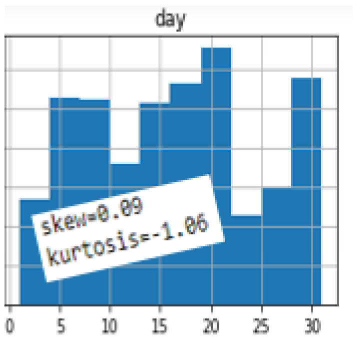


```
dtype=int64, #of nulls=0, #of zeros=3514
Q1=72.0, Q2=72.0, Q3=72.0, IQR=0.0
min=-8019.0, max=102127.0, range=range(-8019, 102127)
mean=1362.27, std=3044.77, median=448.00, mad=1551.51
```

- Negative values indicate costumers who owe the bank money and their balance doesn't cover it
- We introduce *overdraft {0,1}*
  - *1 if balance<0*
- All negative values are replaced with 0
- Binning the balance into 5 equal categories then use their ordinal nature to label encode them

*** the min and max of data is obtained based on training split*

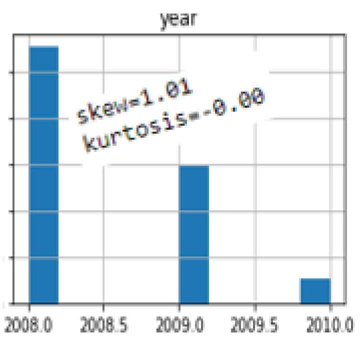
day



```
dtype=int64, #of nulls=0, #of zeros=0
Q1=8.0, Q2=8.0, Q3=8.0, IQR=0.0
min=1.0, max=31.0, range=range(1, 31)
mean=15.81, std=8.32, median=16.00, mad=7.06
```

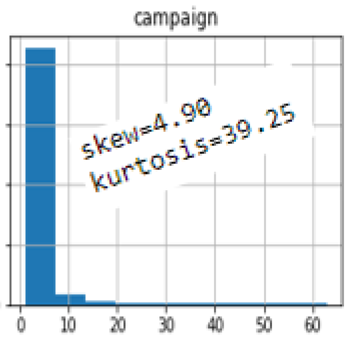
- Although numeric, will illude to favoring higher values over lower values which is not correct.
- Deduce *day_of_week* then hot encode it
- Remove *day* feature

year *{2008,2009,2010}*



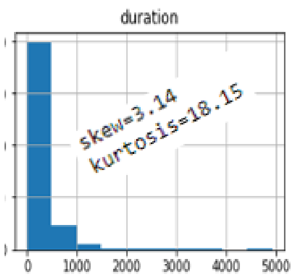
- The ‘year’ is introduced in order to impute the economic indicator correctly. We use the information that the data is ordered from 2008 to 2010 to generate this attribute
- Although numeric, will illude to favoring higher values over lower values which is not correct.
- Hot encode *year*

campaign *[1:36]*



```
dtype=int64, #of nulls=0, #of zeros=0
Q1=1.0, Q2=1.0, Q3=1.0, IQR=0.0
min=1.0, max=63.0, range=range(1, 63)
mean=2.76, std=3.10, median=2.00, mad=1.79
```

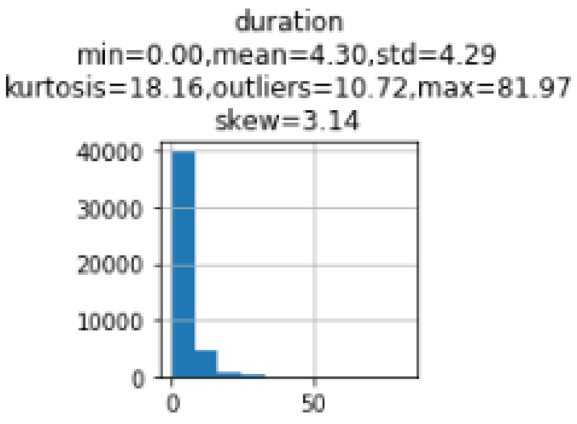
duration



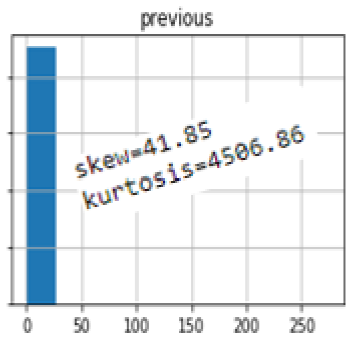
```
dtype=int64, #of nulls=0, #of zeros=3
Q1=103.0, Q2=103.0, Q3=103.0, IQR=0.0
min=0.0, max=4918.0, range=range(0, 4918)
mean=258.16, std=257.53, median=180.00, mad=170.97
```

- Define the variable through minutes instead of seconds. This hides a lot of unnecessary details and maintains all data intact.

- After doing so, the range of the variable become:
  duration [1:81]
  ** further discussed in feature engineering section

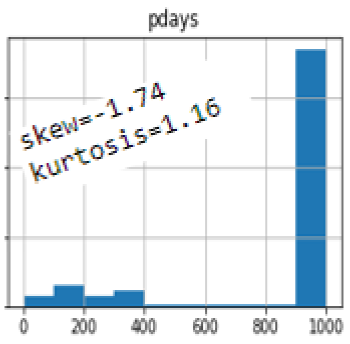


previous *[0,275]*



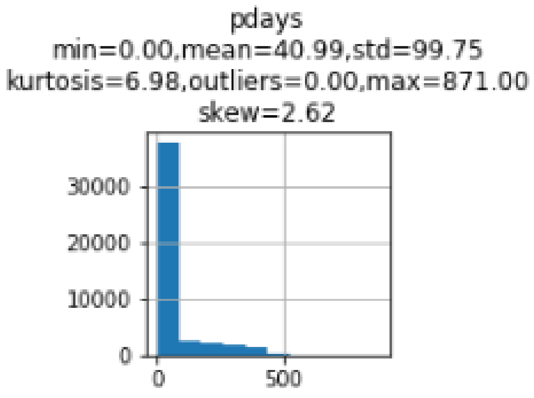
```
dtype=int64, #of nulls=0, #of zeros=36954
Q1=0.0, Q2=0.0, Q3=0.0, IQR=0.0
min=0.0, max=275.0, range=range(0, 275)
mean=0.58, std=2.30, median=0.00, mad=0.95
```

pdays

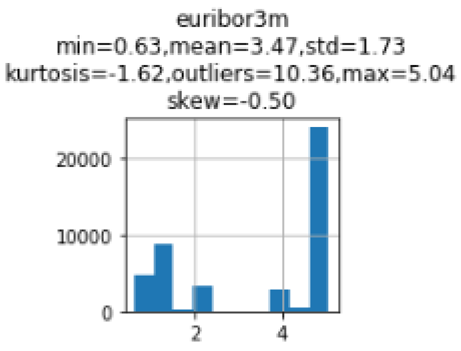


```
dtype=int64, #of nulls=0, #of zeros=0
Q1=999.0, Q2=999.0, Q3=999.0, IQR=0.0
min=1.0, max=999.0, range=range(1, 999)
mean=857.57, std=303.25, median=999.00, mad=231.21
```

- The *pdays* value 999 that represent "not previously contacted for a previous campaign" is over-riding the actual statistics of the variable
- Represent this information with a different value
  pdays=0 → not previously contacted for a previous campaign

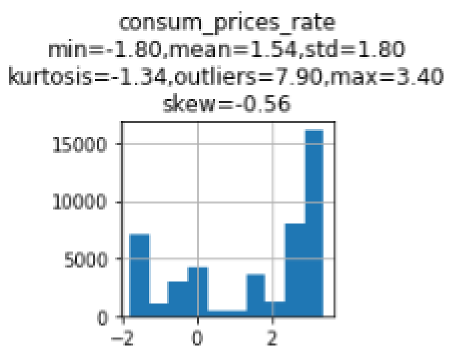


euribor3m
- direct driver for setting interest rates on bank products
- The 0 values in *euribor3m* is intentional, since this value has 3decimal places approximation in its definition.
- However, this is translated as a distinctive value in the predictive models, so we use forwardfilling to replace zeros with the last non-zero value



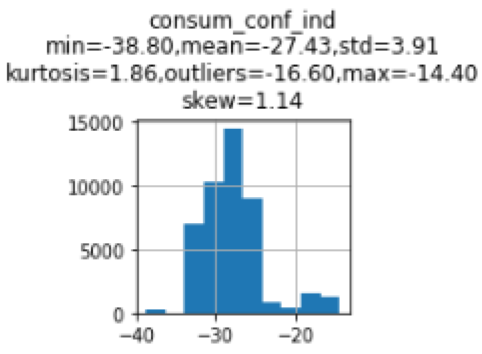
<u>consum_prices_rate</u>

- reflects inflation



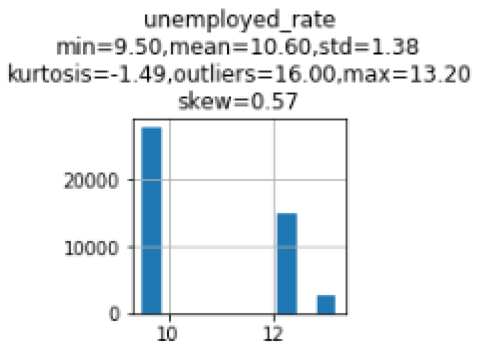
<u>consum_conf_ind</u>

- reflects economic growth

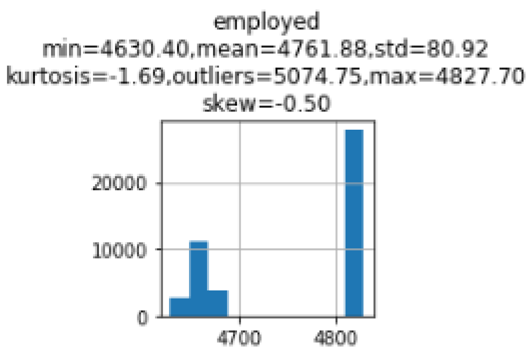


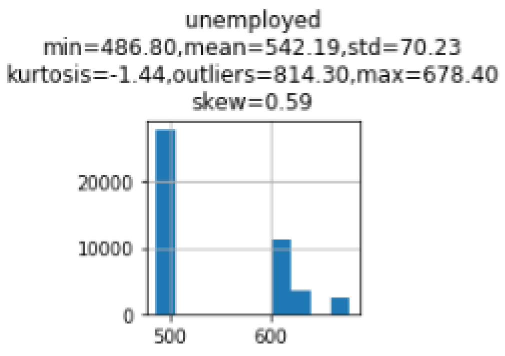
<u>unemployed_rate</u>

- number of unemployed w.r.t total number of adults>18



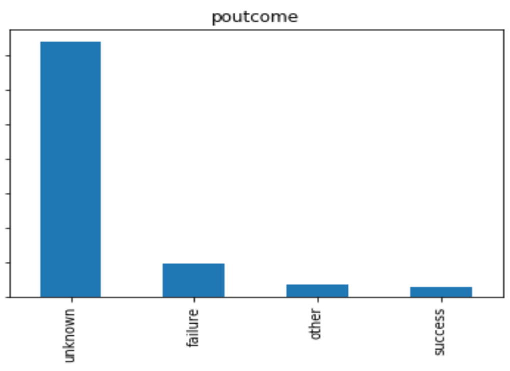
employed,unemployed





b) Categorical variables:

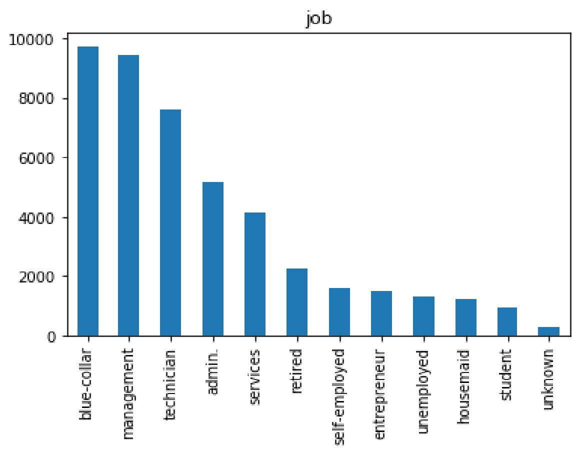
poutcome



```
dtype=object
#of nulls|zeros=0

success(1511)
failure(4901)
other(1840)
unknown(36959)
```

- *poutcome=unknown* & (*pdays=999*| *previous=0*) → *poutcome=non_existant*
- *poutcome=unknown* & *pdays!=999* are only 5 entries → drop the rows
- Introduce ***poutcome_missing***; a binary flag that tells whether *poutcome=other*
- **encode** *poutcome* into *0 (other, failure)* and *1 (success)*

job

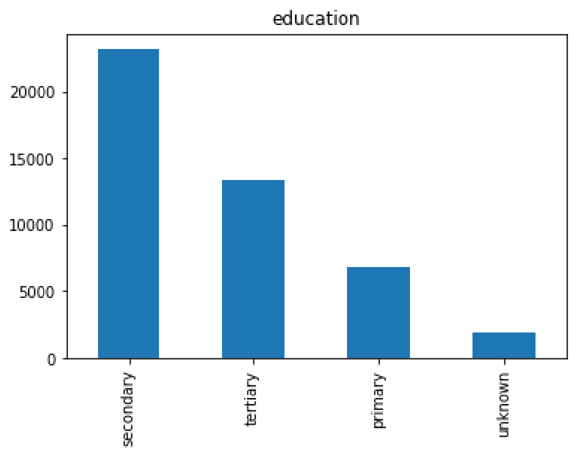


```
dtype=object
#of nulls|zeros=0

management(9458)
blue-collar(9732)
retired(2264)
unknown(288)
technician(7597)
admin.(5171)
unemployed(1303)
services(4154)
student(938)
entrepreneur(1487)
self-employed(1579)
housemaid(1240)
```

- 12 unique values will cause high cardinality if left as is and high dimensionality if hot-encoded
- maintain *unknown* as a class-label
- Introduce *job_missing*; a binary flag that tells whether *job=unknown* or not
- Encode values based on value counts w.r.t y (aka frequency encoding).

** This method of encoding is derived from target encoding, with an important difference that only the training y is taken in consideration to deduce the new values of *job*

education

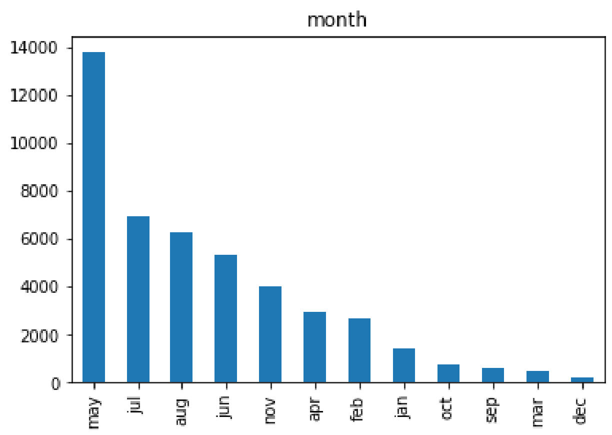


```
dtype=object
#of nulls|zeros=0

secondary(23202)
primary(6851)
tertiary(13301)
unknown(1857)
```

- *primary,secondary,teritary* has a ordinal nature to it, so we use judgment encoding which is the same as label encoding but with contextual logic to the encoding

- sklearn labelencoding encodes classes according to .unique which misses the ordinal nature of the classes, so we opt to specify the classes are encoded as follows: {*unkown:0, primary:1,secondary:2,teritary:3*}
- Introduce *education_missing* is a binary flag that tells whether *education=unknown*

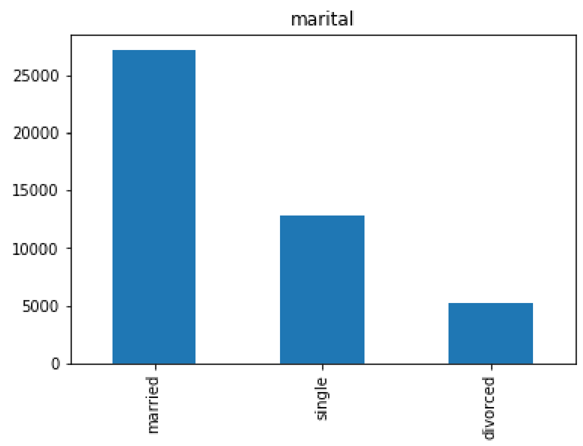
month



```
dtype=object
#of nulls|zeros=0

jun(5341)
sep(579)
aug(6247)
jan(1403)
oct(738)
dec(214)
feb(2649)
jul(6895)
apr(2932)
nov(3970)
mar(477)
may(13766)
```

- Although easily transferred to numeric, will illude to favoring higher values over lower values which is not correct.
- Hot encode *month*

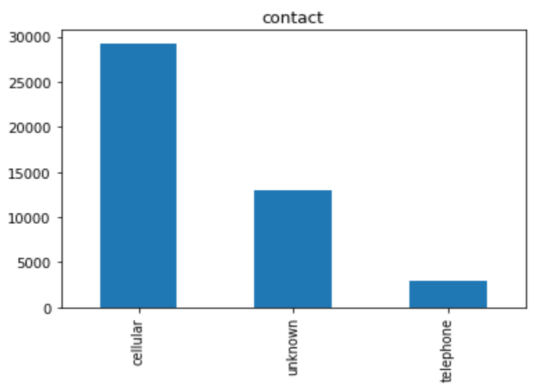
marital



```
dtype=object
#of nulls|zeros=0

divorced(5207)
married(27214)
single(12790)
```

- Has no ordinal nature so can't label encode, hence hot-encoding is used

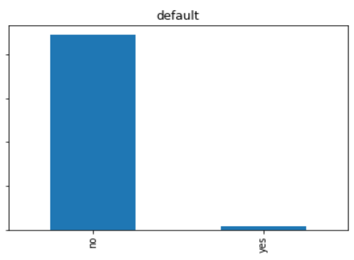
contact

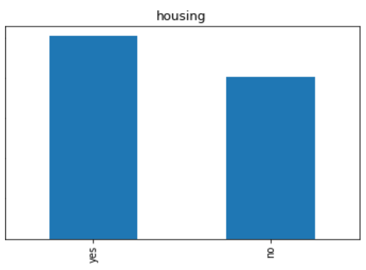


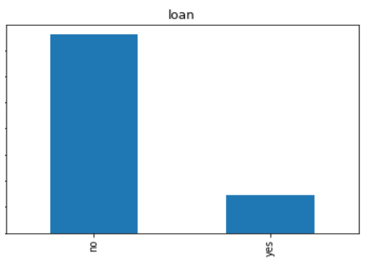
```
dtype=object
#of nulls|zeros=0

cellular(29285)
telephone(2906)
unknown(13020)
```

- Introduce *contact_missing*; a binary flag that tells whether *contact=other*
- Encode *contact* into *0 (other, failure)* and *1 (success)*

default,housing,loan







```
dtype=object
#of nulls|zeros=0

yes(815)
no(44396)
```

```
dtype=object
#of nulls|zeros=0

yes(25130)
no(20081)
```

```
dtype=object
#of nulls|zeros=0

yes(7244)
no(37967)
```

- Binary encode these three features
- We introduce *indebt* feature *[0,2] = default+housing+loan*

  ** further discussed in bivariate analysis section

Section 5: Feature engineering (multi-variate data analysis)

In this section, we attempt to solidify our understanding of the features by making hypotheses and examining the data to validate or negate them. This will be handy when our model concludes to see how it compares. This also gives way to revised transformations and scaling techniques for our features. For this analysis, we keep the analysis strictly limited to the features and do not look into how the features affect the target to avoid un-intentionally manipulating the data to get better results. For better visualization, we consider the data frame after tidying features but before hot-encoding them. For a more unified structure, we represent all features in its numeric form and use Pearson correlation to guide our analysis.

**** DISCLAIMER**

It's crucial to note that EDA is meant to give the bank insights to further tailor their product to costumers' needs and will not be used to alter the features accordingly. Let's take the age of a costumer for example, if you're offering an 80 year old costumer the term deposit product, you don't want to offer years-long tenor to a costumer whose age is beyond the average lifetime of people in Portugal (78 in 2008). But if you have enough pointers to how costumers of this age approach their finances, you can approach him with a more agreeable plan for that phase of his life without risking investing in the wrong costumer. So this EDA attempts to relate features w.r.t each other not with the target; to avoid compromising the data fed to the predictive model, especially since the target data is severely imbalanced so insights are not conclusive for generalization.

The features can be divided into:

a) Personal info & personal standing of costumers
b) Current campaign communication with costumers
c) Previous contact with costumers
d) Date dependent economic and social indicators

a) Personal info & personal standing:

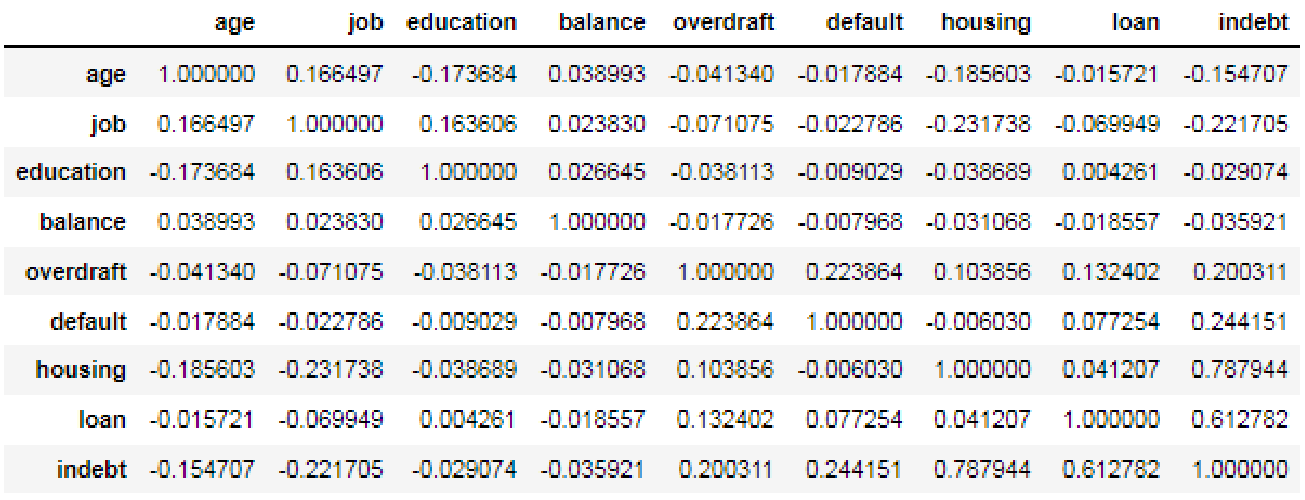
From a bank's point of view, the ultimate goal is to target the list of costumers that are most probable to invest in this product. Hence, we attempt to relate costumers' personal data with their standing with the bank. To facilitate this analysis, we propose binning the age of costumers into *age_grp* of three groups:

youth: up to 25, adult 25 to 65 and old from 65 up. These groups were chosen according to the information that the pension age in Portugal in 2008 was 65 and that it's best to obtain house mortgages starting 25 years. We also propose having a collective representative for all loans called *indebt* that simply sums the features *default,housing,loan*. These 3 features represent different indicators of loans that might affect each other but are not derivable from each other. The following table shows the dependencies inflicted on the data during feature engineering and transformation. Each of the following hypotheses deal with features from different columns.

personal info → *age, job, marital, education*

personal standing related variables → *default, balance, overdraft, housing, loan*

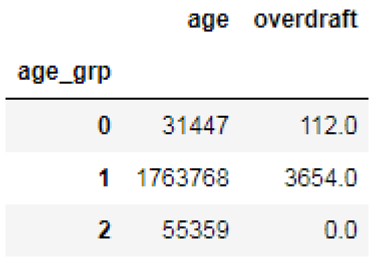
Table of dependencies, bivariate analysis is done between variables of different columns to avoid inherent dependency						
education	*age_grp*	*job*	*indebt*	*marital*	*overdraft*	*balance*
education_missing	*age*	*job_missing*	*default, housing, loan*			



	age	job	education	balance	overdraft	default	housing	loan	indebt
age	1.000000	0.166497	-0.173684	0.038993	-0.041340	-0.017884	-0.185603	-0.015721	-0.154707
job	0.166497	1.000000	0.163606	0.023830	-0.071075	-0.022786	-0.231738	-0.069949	-0.221705
education	-0.173684	0.163606	1.000000	0.026645	-0.038113	-0.009029	-0.038689	0.004261	-0.029074
balance	0.038993	0.023830	0.026645	1.000000	-0.017726	-0.007968	-0.031068	-0.018557	-0.035921
overdraft	-0.041340	-0.071075	-0.038113	-0.017726	1.000000	0.223864	0.103856	0.132402	0.200311
default	-0.017884	-0.022786	-0.009029	-0.007968	0.223864	1.000000	-0.006030	0.077254	0.244151
housing	-0.185603	-0.231738	-0.038689	-0.031068	0.103856	-0.006030	1.000000	0.041207	0.787944
loan	-0.015721	-0.069949	0.004261	-0.018557	0.132402	0.077254	0.041207	1.000000	0.612782
indebt	-0.154707	-0.221705	-0.029074	-0.035921	0.200311	0.244151	0.787944	0.612782	1.000000

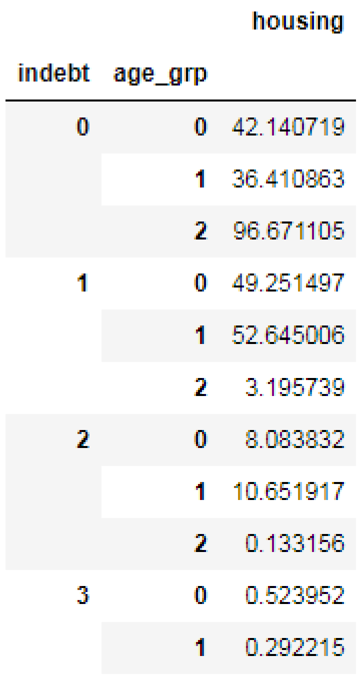
Hypotheses & Findings:

- Old people have no overdraft
  #df.groupby(['age_grp'])[['age','overdraft']].sum()



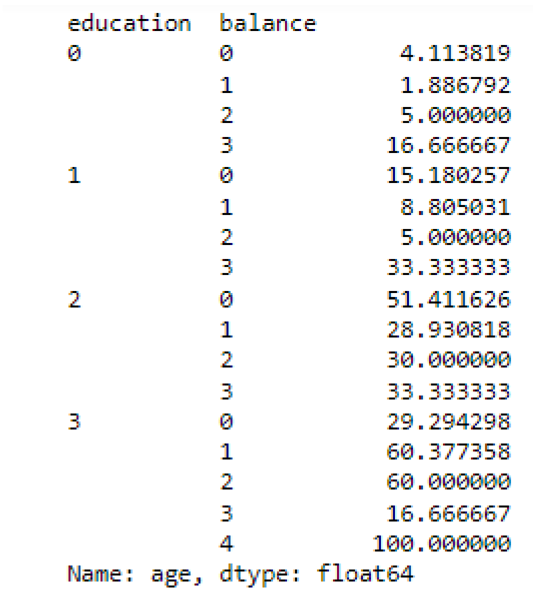
	age	overdraft
age_grp		
0	31447	112.0
1	1763768	3654.0
2	55359	0.0

- 96% of old people in the dataset have paid all their loans
  #df.groupby(['indebt','age_grp'])['housing'].count()*100/df.groupby(['age_grp'])['housing'].count()

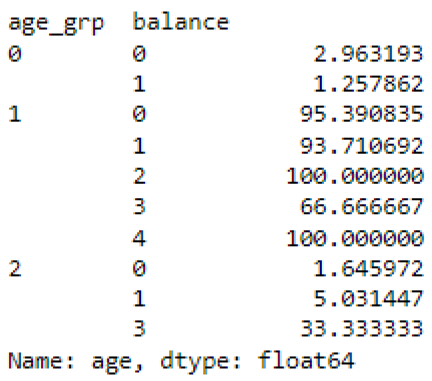


		housing
indebt	**age_grp**	
0	**0**	42.140719
	1	36.410863
	2	96.671105
1	**0**	49.251497
	1	52.645006
	2	3.195739
2	**0**	8.083832
	1	10.651917
	2	0.133156
3	**0**	0.523952
	1	0.292215

- The highest balances are associated with adults who are tertiary educated, only
  #df.groupby(['education','balance'])['age'].count()*100/df.groupby(['balance'])['age'].count()
  #df.groupby(['age_grp','balance'])['age'].count()*100/df.groupby(['balance'])['age'].count()



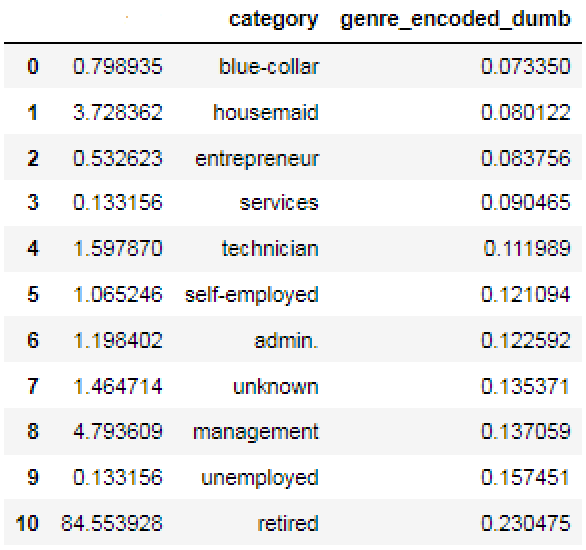
```
education  balance
0          0           4.113819
           1           1.886792
           2           5.000000
           3          16.666667
1          0          15.180257
           1           8.805031
           2           5.000000
           3          33.333333
2          0          51.411626
           1          28.930818
           2          30.000000
           3          33.333333
3          0          29.294298
           1          60.377358
           2          60.000000
           3          16.666667
           4         100.000000
Name: age, dtype: float64
```



```
age_grp  balance
0        0            2.963193
         1            1.257862
1        0           95.390835
         1           93.710692
         2          100.000000
         3           66.666667
         4          100.000000
2        0            1.645972
         1            5.031447
         3           33.333333
Name: age, dtype: float64
```

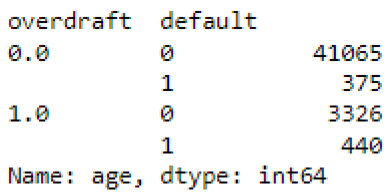
- 84% of Old people are retired, then with a landslide the most occupied jobs are management and housemaids. Services that require manual labor comes at the bottom of the list as expected

#df.query('age_grp==2').groupby(['job']).count()[['age']]*100/len(df.query('age_grp==2'))



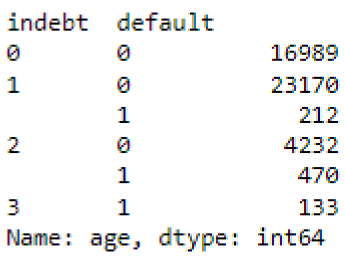
		category	genre_encoded_dumb
0	0.798935	blue-collar	0.073350
1	3.728362	housemaid	0.080122
2	0.532623	entrepreneur	0.083756
3	0.133156	services	0.090465
4	1.597870	technician	0.111989
5	1.065246	self-employed	0.121094
6	1.198402	admin.	0.122592
7	1.464714	unknown	0.135371
8	4.793609	management	0.137059
9	0.133156	unemployed	0.157451
10	84.553928	retired	0.230475

- The hypothesis was that People with no default have no overdrafts, but I was overlooking two factors. The first is that there are [two types of overdraft](), and while one type is related to missing payment due dates of loans, there's another one called agreed overdrafts similar to spending money from your credit card that you don't have at the moment. The second is that ,for example, someone not missing his payment's due dates doesn't mean he doesn't use credit cards at all
  # df.groupby(['overdraft','default'])['age'].count()



```
overdraft  default
0.0        0          41065
           1            375
1.0        0           3326
           1            440
Name: age, dtype: int64
```

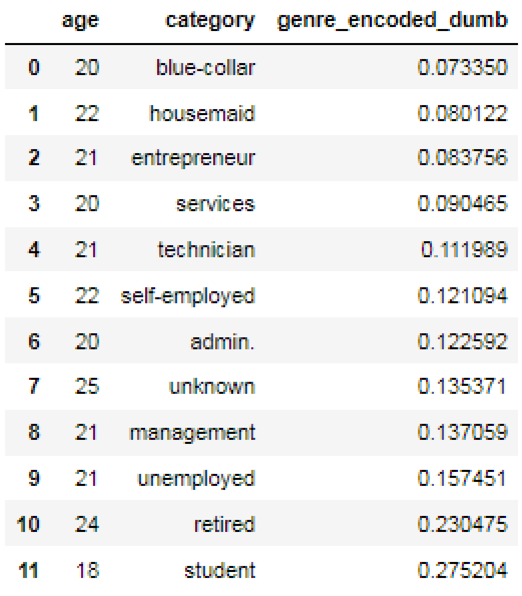
- While we're on default, people with no debts might have no default. This is based on the correlation between default and all types of loans.
  # df.groupby([‘indebt’,’default’])[‘age’].count()



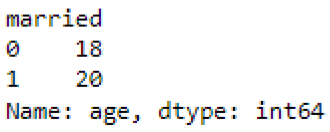
```
indebt  default
0       0          16989
1       0          23170
        1            212
2       0           4232
        1            470
3       1            133
Name: age, dtype: int64
```

- The minimum age for any job available in the dataset is 20 years, ages from 18 up to this age can only be students and not so coincidentally, 20 years is also the smallest married costumer in the dataset
  #df.groupby(['job'])['age'].min()

#df.groupby(['married']).min().age

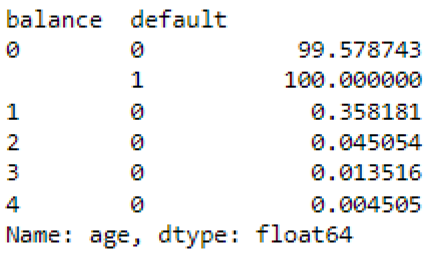


	age	category	genre_encoded_dumb
0	20	blue-collar	0.073350
1	22	housemaid	0.080122
2	21	entrepreneur	0.083756
3	20	services	0.090465
4	21	technician	0.111989
5	22	self-employed	0.121094
6	20	admin.	0.122592
7	25	unknown	0.135371
8	21	management	0.137059
9	21	unemployed	0.157451
10	24	retired	0.230475
11	18	student	0.275204



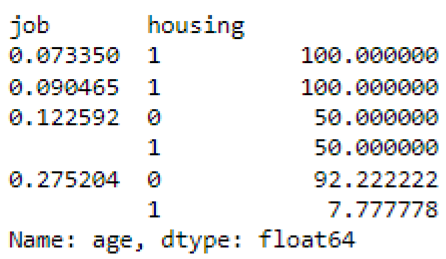
```
married
0    18
1    20
Name: age, dtype: int64
```

- Costumer's who have default usually have small balances that doesn't suffice paying the installments
  #df.groupby(['balance','default'])['age'].count()*100/df.groupby(['default'])['age'].count()

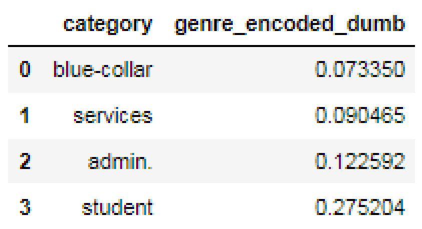


```
balance  default
0        0           99.578743
         1          100.000000
1        0            0.358181
2        0            0.045054
3        0            0.013516
4        0            0.004505
Name: age, dtype: float64
```

- Only 7% of students below 21 years have housing loans
  #df.query('age<21').groupby(['job','housing'])['age'].count()*100/df.query('age<21').groupby([
  'job'])['age'].count()

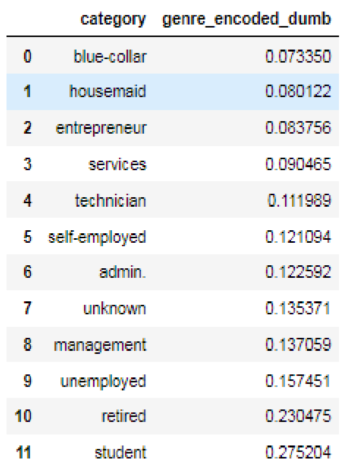


```
job       housing
0.073350  1          100.000000
0.090465  1          100.000000
0.122592  0           50.000000
          1           50.000000
0.275204  0           92.222222
          1            7.777778
Name: age, dtype: float64
```

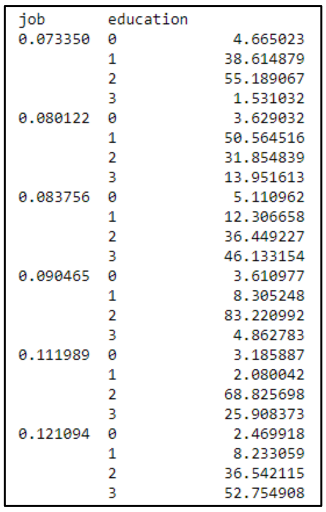


	category	genre_encoded_dumb
0	blue-collar	0.073350
1	services	0.090465
2	admin.	0.122592
3	student	0.275204

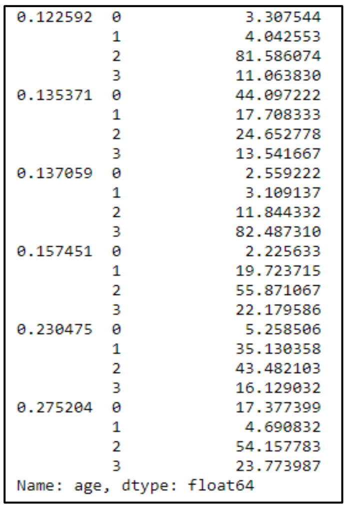
- You can tell the required education for each job using value counts or mode
  - ➔ Managerial, self-employed and entrepreneurs mostly hold tertial education
  - ➔ Admin, blue-collar, services, technician need secondary education
  - ➔ The only job that suffice with primary education is housemaid



	category	genre_encoded_dumb
0	blue-collar	0.073350
1	housemaid	0.080122
2	entrepreneur	0.083756
3	services	0.090465
4	technician	0.111989
5	self-employed	0.121094
6	admin.	0.122592
7	unknown	0.135371
8	management	0.137059
9	unemployed	0.157451
10	retired	0.230475
11	student	0.275204



```
job       education
0.073350  0             4.665023
          1            38.614879
          2            55.189067
          3             1.531032
0.080122  0             3.629032
          1            50.564516
          2            31.854839
          3            13.951613
0.083756  0             5.110962
          1            12.306658
          2            36.449227
          3            46.133154
0.090465  0             3.610977
          1             8.305248
          2            83.220992
          3             4.862783
0.111989  0             3.185887
          1             2.080042
          2            68.825698
          3            25.908373
0.121094  0             2.469918
          1             8.233059
          2            36.542115
          3            52.754908
```



```
0.122592  0             3.307544
          1             4.042553
          2            81.586074
          3            11.063830
0.135371  0            44.097222
          1            17.708333
          2            24.652778
          3            13.541667
0.137059  0             2.559222
          1             3.109137
          2            11.844332
          3            82.487310
0.157451  0             2.225633
          1            19.723715
          2            55.871067
          3            22.179586
0.230475  0             5.258506
          1            35.130358
          2            43.482103
          3            16.129032
0.275204  0            17.377399
          1             4.690832
          2            54.157783
          3            23.773987
Name: age, dtype: float64
```

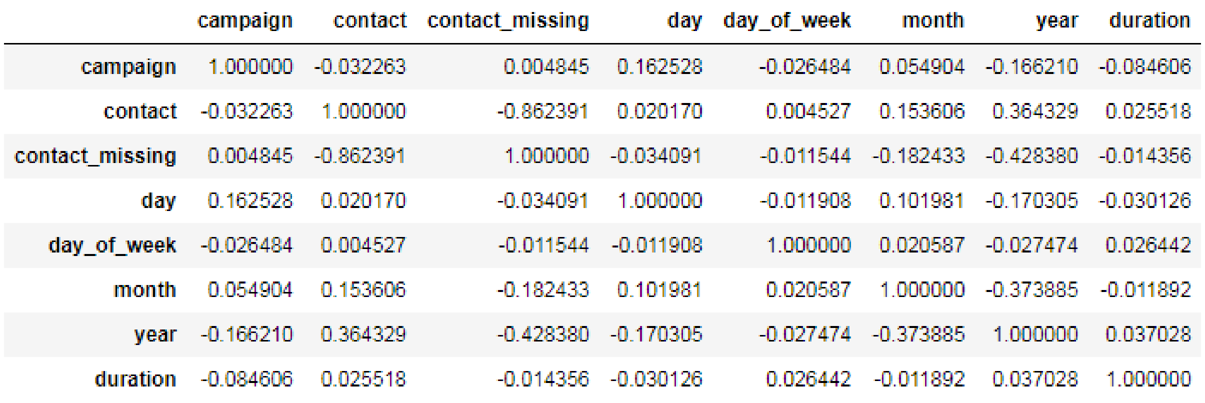
- The observations made throughout this section solidify that there is an inherent cut in the age feature at ages 25 and 65.

b) <u>Current campaign communication with costumers</u>

This type of data governs how the bank communicate with each costumer during the campaign at hand. The observations done on this data is expected to focus on how well costumers respond to methods of reaching out from the bank. Do people require less convincing around the time their salaries become deposited or just before that when cash is tight? Do they respond better around the holidays? Are frequent contacts with the same costumer related to less duration of contact? Is there a favorable method of contact?

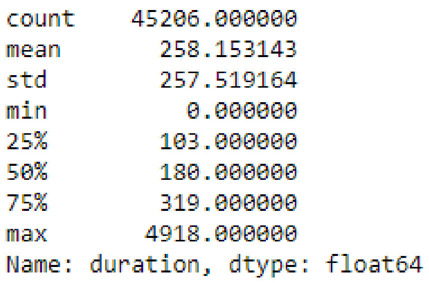
this_campaign_comm → *campaign, contact, contact_missing, day, day_of_week, month, year, duration.*

Table of dependencies, bivariate analysis is done between variables of different columns					
contact	*duration*	*day*	*campaign*	*month*	*year*
contact_missing		*day_of_week*			

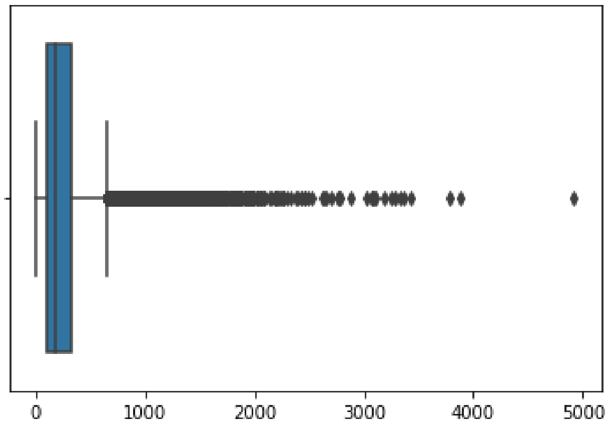


	campaign	contact	contact_missing	day	day_of_week	month	year	duration
campaign	1.000000	-0.032263	0.004845	0.162528	-0.026484	0.054904	-0.166210	-0.084606
contact	-0.032263	1.000000	-0.862391	0.020170	0.004527	0.153606	0.364329	0.025518
contact_missing	0.004845	-0.862391	1.000000	-0.034091	-0.011544	-0.182433	-0.428380	-0.014356
day	0.162528	0.020170	-0.034091	1.000000	-0.011908	0.101981	-0.170305	-0.030126
day_of_week	-0.026484	0.004527	-0.011544	-0.011908	1.000000	0.020587	-0.027474	0.026442
month	0.054904	0.153606	-0.182433	0.101981	0.020587	1.000000	-0.373885	-0.011892
year	-0.166210	0.364329	-0.428380	-0.170305	-0.027474	-0.373885	1.000000	0.037028
duration	-0.084606	0.025518	-0.014356	-0.030126	0.026442	-0.011892	0.037028	1.000000

duration is the only continuous variable related to the campaign's communication with this costumer. We look into its properties to reach an appropriate way to represent it. Its high variance is not conclusive to derive insights. It's also expected that there are outliers since the difference between 75% and max is huge, we validate this with the box plot that confirms the hypothesis.



```
count    45206.000000
mean       258.153143
std        257.519164
min          0.000000
25%        103.000000
50%        180.000000
75%        319.000000
max       4918.000000
Name: duration, dtype: float64
```

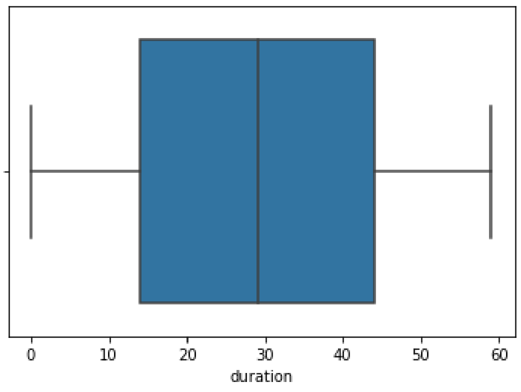


Looking into clipping or cupping the upper outliers, we look into the amount of data that we would be losing. Assuming we're looking into the 99% percentile, there's a lot of entries to lose.

```
durations statistics
upper limit=1269.0
max=4918
len(> upper_lim)=452
```

So instead of removing outliers, we attempt to represent the variable in a more generic manner. This is straight forward through defining the call duration through minutes instead of seconds. This hides a lot of unnecessary details and maintains all data intact.

```
<matplotlib.axes._subplots.AxesSubplot at 0xbc691c8
```



** It's crucial to look at duration from a contextual point of view as well. The duration of a call with a costumer is not data that would bs used beforehand. But rather after a call, when it's already obvious whether the costumer went for the deposit product or not.

We look at the following statistics year by year to discuss overall governing relation without overanalyzing details that might be coincidental over a smaller time interval.

- Method of contact per year
  The relatively high correlation between contact and year gauges us to look into it. It makes sense that the contact method moves toward cellular as years pass by.

```
year    contact
2008.0  0          50.863717
        1          49.136283
2009.0  0           8.769096
        1          91.230904
2010.0  0          19.824293
        1          80.175707
Name: contact, dtype: float64
```

- Costumers contacted per year

```
year
2008.0    27729
2009.0    14859
2010.0     2618
Name: age, dtype: int64
```

It’s interesting that the data starts in 2008 in May, and yet the number of costumers contacted makes up almost 60 % of the data. Let’s note that 2008 is right in the middle of the recession period that lasted till June 2009. So we add a variable *rec* that is a binary representation of 1 untill June 2009 then is 0

- Mode of *duration* and *mean* campaign per *year* and per *rec* variables

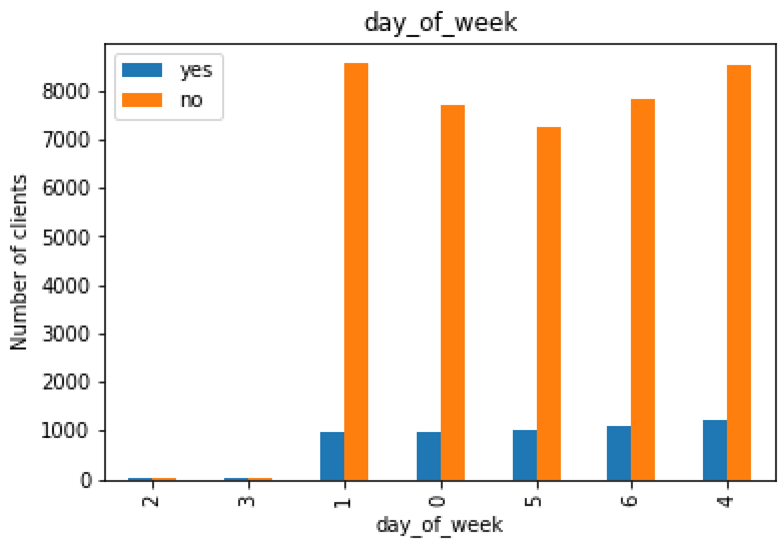
	duration	campaign
rec		
0	5	1.835350
1	13	2.891276

	duration	campaign
year		
2008.0	13	3.186447
2009.0	11	2.130291
2010.0	5	1.884263

It makes sense that the frequency of contacting costumers, whether we’re talking about number of calls or the durations, is highest at the peak of the recession because it was an absolutely necessary product given the bad economy .

- Mode of *day_of_week*

It looks like bank employees focus their communication with costumers away from the middle of the week.

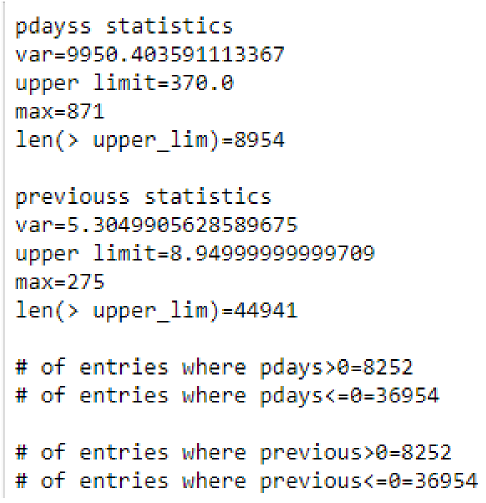


c) <u>Previous contact with costumers</u>

previous_contact → pdays, previous, poutcome, poutcome_missing

The data in this group is particularly challenging to read into because

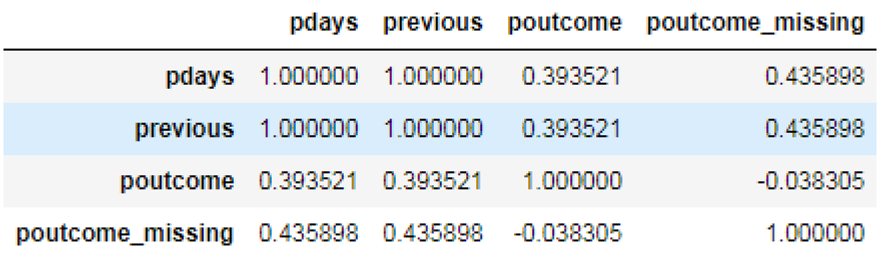
1) There's strong dependency between all variables since we added 'non_existant' value to poutcome based on pdays and previous.
2) Both pdays and previous are continuous variables.



```
pdayss statistics
var=9950.403591113367
upper limit=370.0
max=871
len(> upper_lim)=8954

previouss statistics
var=5.304990562858 9675
upper limit=8.94999999999709
max=275
len(> upper_lim)=44941

# of entries where pdays>0=8252
# of entries where pdays<=0=36954

# of entries where previous>0=8252
# of entries where previous<=0=36954
```

The statistics show lots of outliers in *previous* and huge variance in *pdays*! As seen from the statistics, truncating outliers will cause a huge loss in information and binning it will be arbitrary as well. At this point, we're inclined to changing at least one variable -say pdays- to a binary indicator that's 0 when the original data is 0 and 1 otherwise. Looking into the number of entries we're giving up numeric details in, we find that the number of those is still significantly lower than what we'd lost if we truncate outliers.

Attempt: change pdays to a binary indicator: 0 if pdays=0 , 1 otherwise



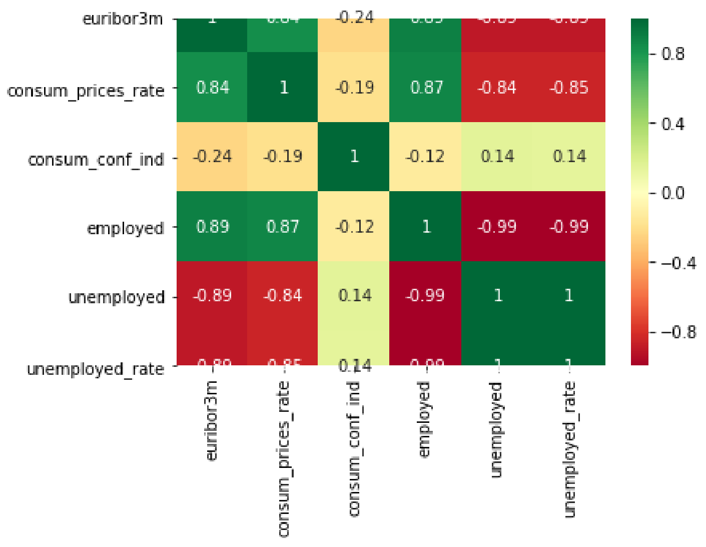
	pdays	previous	poutcome	poutcome_missing
pdays	1.000000	1.000000	0.393521	0.435898
previous	1.000000	1.000000	0.393521	0.435898
poutcome	0.393521	0.393521	1.000000	-0.038305
poutcome_missing	0.435898	0.435898	-0.038305	1.000000

After using binary values to represent pdays, we find that its information is completely represented in previous

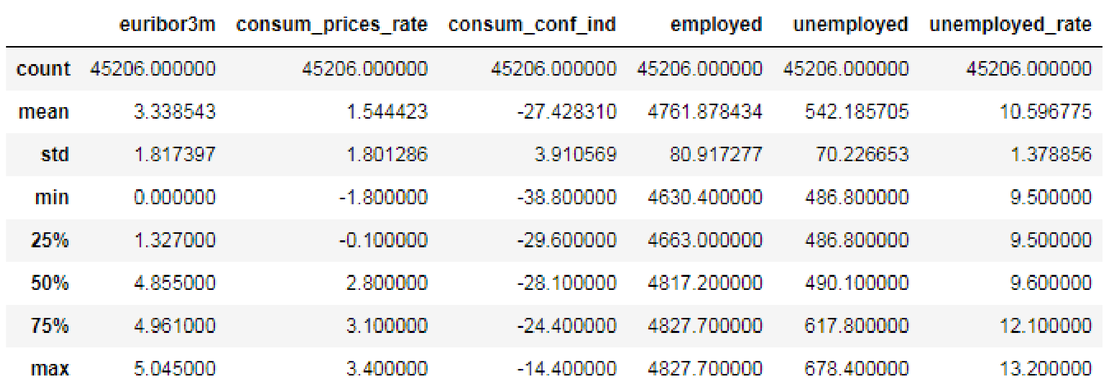
Therefore, drop pdays

d) <u>Date dependent economic and social indicators</u>

eco_indic→ *euribor3m, consum_prices_rate, consum_conf_ind, employed, unemployed, unemployed_rate*



From the correlation matrix alone, it's obvious that the 3 variables *employed,unemployed,unemployed_rate* are almost completely dependent. We choose to keep all three even though the underlying understanding of the features would gauge us to use only one.



	euribor3m	consum_prices_rate	consum_conf_ind	employed	unemployed	unemployed_rate
count	45206.000000	45206.000000	45206.000000	45206.000000	45206.000000	45206.000000
mean	3.338543	1.544423	-27.428310	4761.878434	542.185705	10.596775
std	1.817397	1.801286	3.910569	80.917277	70.226653	1.378856
min	0.000000	-1.800000	-38.800000	4630.400000	486.800000	9.500000
25%	1.327000	-0.100000	-29.600000	4663.000000	486.800000	9.500000
50%	4.855000	2.800000	-28.100000	4817.200000	490.100000	9.600000
75%	4.961000	3.100000	-24.400000	4827.700000	617.800000	12.100000
max	5.045000	3.400000	-14.400000	4827.700000	678.400000	13.200000

Summary of EDA:

- Encode all categorical values into numerical values using hot encoding, label encoding and target-derived encoding
- Impute *year,day_of_week* and the economic indicators
- Add *indebt*
- Bin *age* into *age_grp*
- Bin *balance* into 5 ordinal equally divided groups
- Add *rec* to represent the recession period
- Drop *pday,day*
- Using minutes representation for *duration*

Section 6: Problems with the dataset

After performing raw data exploration in the previous section. It's time we mention all the issues we found during our analysis.

a) Missing data: #sections:4&5

Although there are no NaNs in our data, there are ambiguous values such as *unknown,other* in the features: *education,poutcome,contact*. We use binary indicators to preserve the information of ambiguous values and then encode it within the well-defined values of each feature

b) High cardinality: #sections:4&5

This issue is usually associated with categorical features. This problem appears in *job* feature. Using this kind of data to train a predictive model results in non-conclusive models that fail to generalize well. The same issue is caused having numerical features with high variance such as *balance,duration*. We opt for binning balance into 5 equally divided groups. We leave duration as is since this variable, specifically, should not be included in the prediction.

c) Nominal & ordinal nature of data: #sections:4&5

According to this information, the type of encoding differs. *education* has an ordinal nature since secondary come after primary and so forth, so numbering them makes sense. This is unlike *marital* or *job* or *month,year* where there is no relation between their values except that they belong to the same category, in this case target-derived encoding or hot encoding are used.

d) High dimensionality: #sections:4&5

We are mindful of not increasing the number of features unnecessarily, whether through dropping features that are well-represented by other features, or being picky towards the type of encoding used for features. Hot encoding every variable would end up with a huge list of binary features that are hardly influential to the predictive models.

e) Skewed-data #section:7

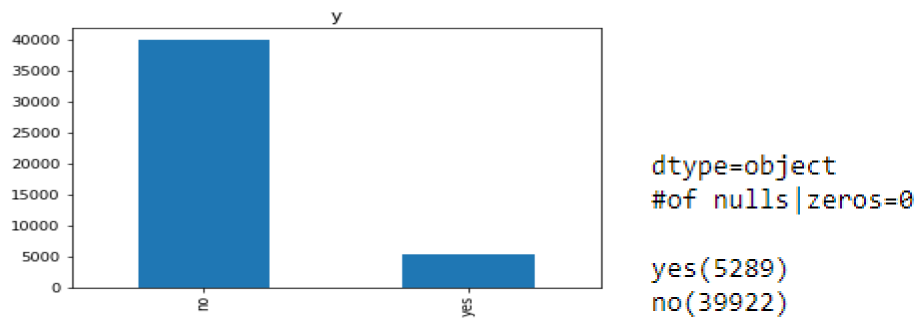
In regard of categorical data, we attempt to transform those features into numerical features that start at 0. This inherently removes a big chunk of skewness. This is done through binning and encoding of features. Another well-known method used to handle this is log-transformation that is used to conform to having positive values across all features and attempt to approach normally-distribution. The latter method is applied in the next section.

f) Outliers: #section:7

Although we’ve discussed the prominent existence of outliers in the features, we haven’t actually handled them yet. To handle outliers, percentile is used. Lower outliers are considered below the 0.1 percentile and higher outliers are beyond the 0.99 percentile. You can choose to clip the features before the 0.1 percentile and after the 0.99 percentile or cap the features to these values or scale the features using statistics of values above the 0.1 percentile and below the 0.99 percentile. You could also opt to not drop outliers at all. This is a case-by-case choice that will be discussed in the next section

g) Class imbalance #section:8

The class imbalance in our problem is obvious. This causes the predictive model to predict ‘no’ more often because it’s the majority class.



```
dtype=object
#of nulls|zeros=0

yes(5289)
no(39922)
```

To combat class imbalance problem:

1) We avoid using statistics that involve the positive and negative classes together since it’ll be misleading such as accuracy. We also avoid using ROC curve; although it’s preferred in case of imbalanced classes, it’s very sensitive to skewness that could cause a huge change in the curve We will use **precision** to evaluate our models.
2) We also make sure to stratify data between testing and training to maintain the same class distribution among both sets.
3) Hyperparameter tuning of models to balance-out this problem

Section 7: scaling and transformation

Numeric encoded (originally categoric) features → Do not scale or transform this data. Attempting either on these features will make the features lose their categoricity.

Numeric (economic indicators) features → Transform but do not attempt to remove outliers. Extreme values in this case are all meaningful and crucial to the conclusion

Numeric (continuous) features → Transform and scale as needed

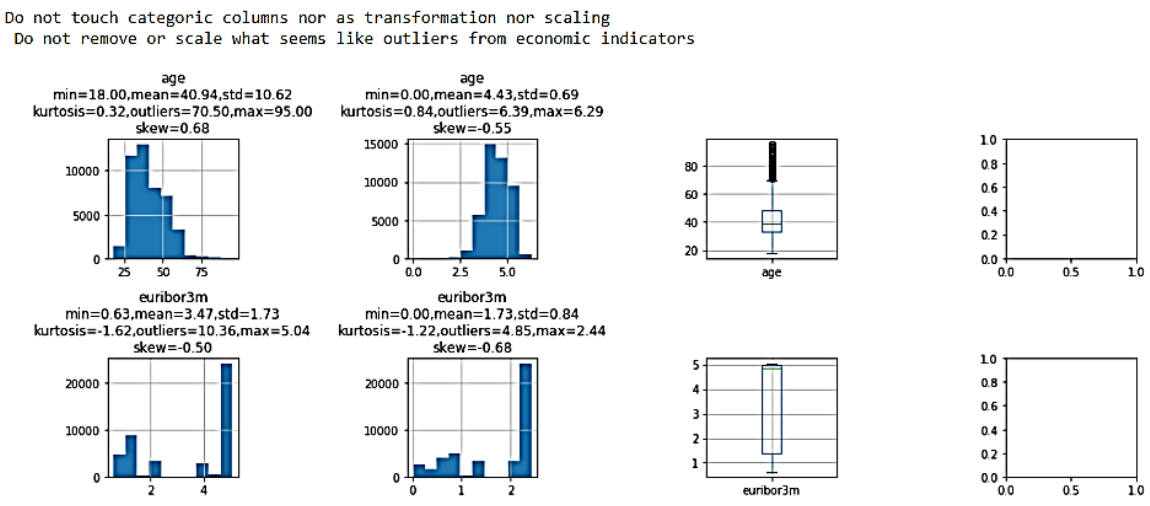
To combat skewness → log-transform

- To avoid losing negative values, we need all feature values to be at least 1, so we use this formula: df[n]=df[n]-(df[n].min()-1)
- Take log transformation of df[n]

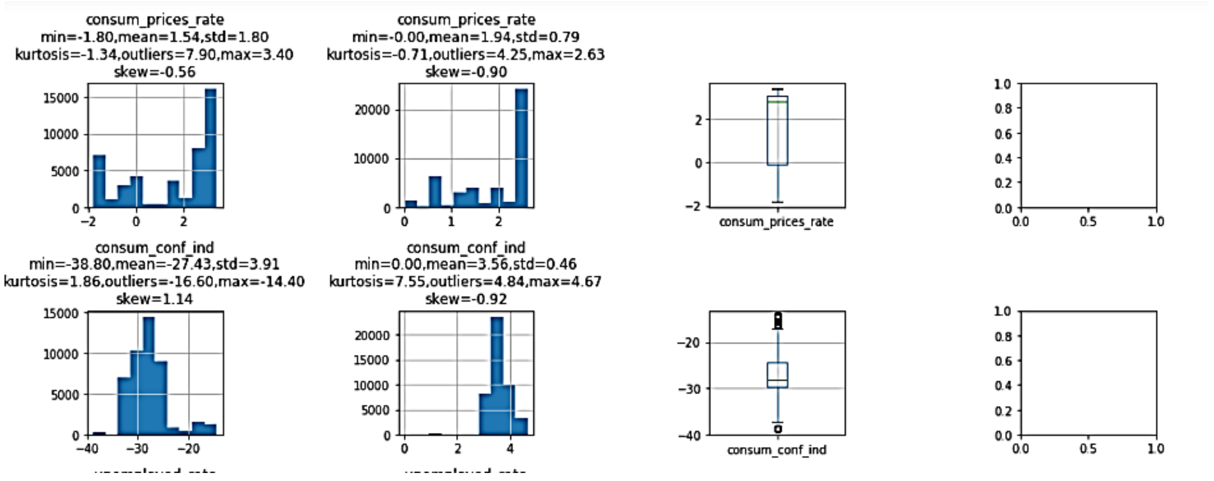
To combat outliers → robust scaler

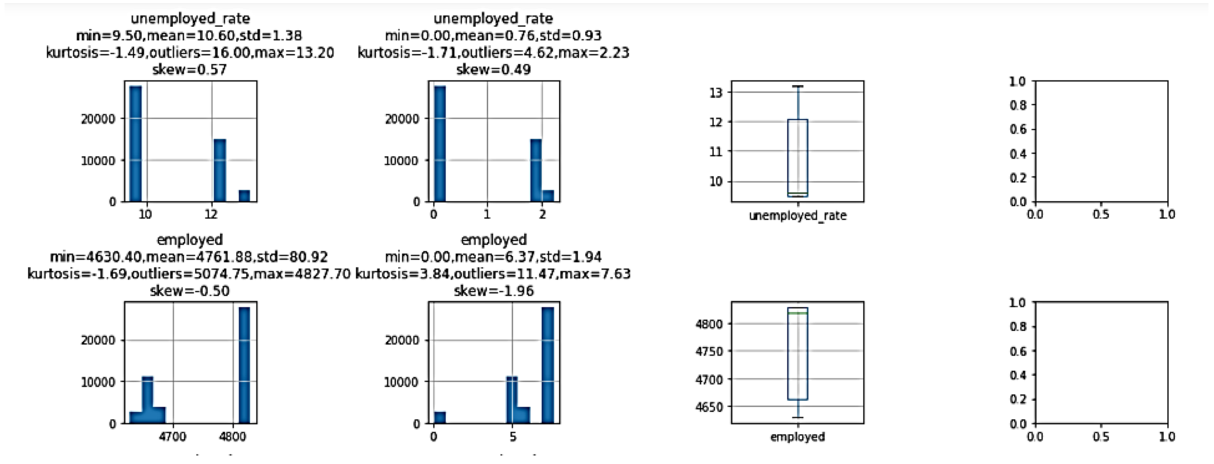
- To pick the variables that require this scaling, we use the famous IQR method where the cut-off limits are Q1 - 1.5 * IQR and Q3 + 1.5 * IQR.
  *** If (df[n].max()>Q3 + 1.5 * IQR) or (df[n].min()<Q1 - 1.5 * IQR) → scale*

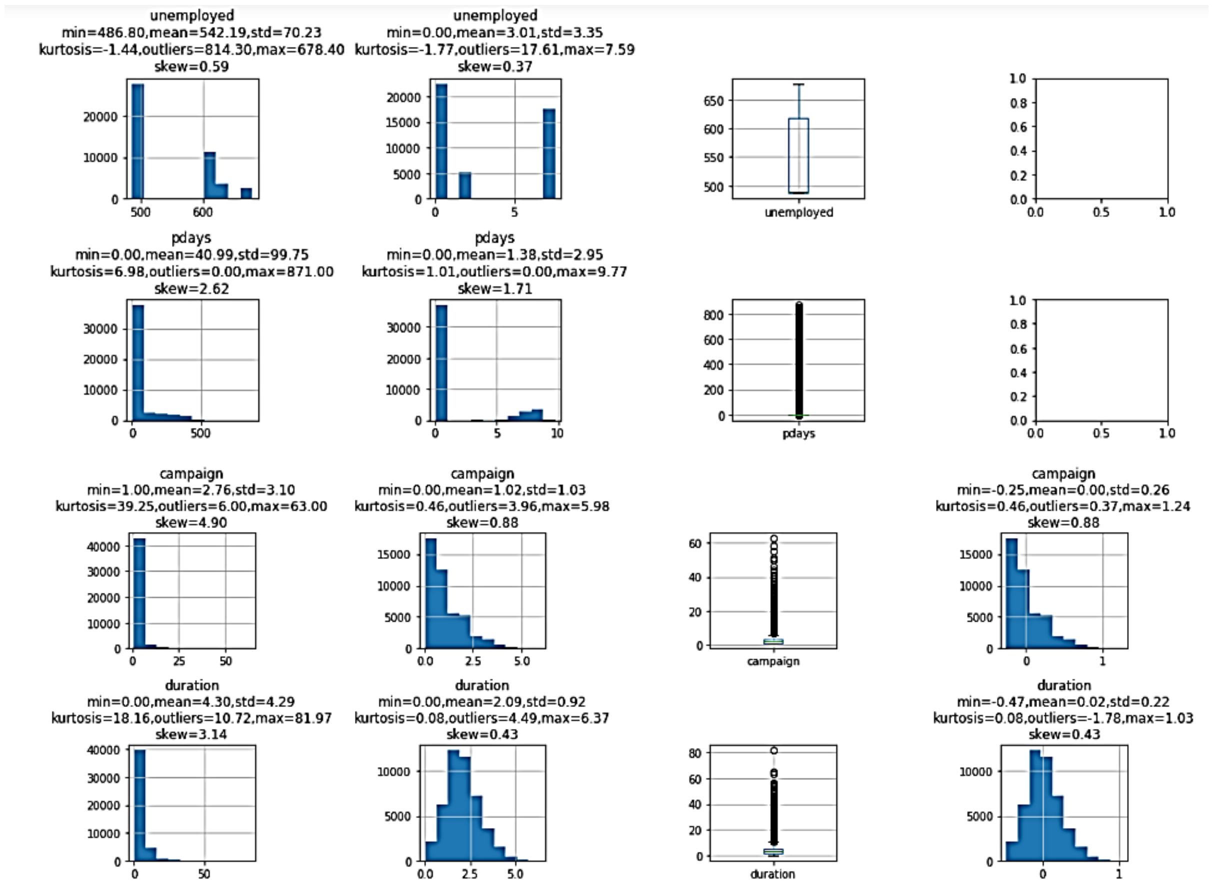
- This scaler utilizes the centering and scaling on data within the defined percentile (0.1,0.99). This isolates the effect of outliers on the scaled data and maintains the same distribution.



```
Do not touch categoric columns nor as transformation nor scaling
 Do not remove or scale what seems like outliers from economic indicators
```


consum_prices_rate
min=-1.80,mean=1.54,std=1.80
kurtosis=-1.34,outliers=7.90,max=3.40
skew=-0.56
consum_prices_rate
min=-0.00,mean=1.94,std=0.79
kurtosis=-0.71,outliers=4.25,max=2.63
skew=-0.90
consum_prices_rate
consum_conf_ind
min=-38.80,mean=-27.43,std=3.91
kurtosis=1.86,outliers=-16.60,max=-14.40
skew=1.14
consum_conf_ind
min=0.00,mean=3.56,std=0.46
kurtosis=7.55,outliers=4.84,max=4.67
skew=-0.92
consum_conf_ind


unemployed_rate
min=9.50,mean=10.60,std=1.38
kurtosis=-1.49,outliers=16.00,max=13.20
skew=0.57
unemployed_rate
min=0.00,mean=0.76,std=0.93
kurtosis=-1.71,outliers=4.62,max=2.23
skew=0.49
unemployed_rate
employed
min=4630.40,mean=4761.88,std=80.92
kurtosis=-1.69,outliers=5074.75,max=4827.70
skew=-0.50
employed
min=0.00,mean=6.37,std=1.94
kurtosis=3.84,outliers=11.47,max=7.63
skew=-1.96
employed


unemployed
min=486.80,mean=542.19,std=70.23
kurtosis=-1.44,outliers=814.30,max=678.40
skew=0.59
unemployed
min=0.00,mean=3.01,std=3.35
kurtosis=-1.77,outliers=17.61,max=7.59
skew=0.37
unemployed
pdays
min=0.00,mean=40.99,std=99.75
kurtosis=6.98,outliers=0.00,max=871.00
skew=2.62
pdays
min=0.00,mean=1.38,std=2.95
kurtosis=1.01,outliers=0.00,max=9.77
skew=1.71
pdays
campaign
min=1.00,mean=2.76,std=3.10
kurtosis=39.25,outliers=6.00,max=63.00
skew=4.90
campaign
min=0.00,mean=1.02,std=1.03
kurtosis=0.46,outliers=3.96,max=5.98
skew=0.88
campaign
campaign
min=-0.25,mean=0.00,std=0.26
kurtosis=0.46,outliers=0.37,max=1.24
skew=0.88
duration
min=0.00,mean=4.30,std=4.29
kurtosis=18.16,outliers=10.72,max=81.97
skew=3.14
duration
min=0.00,mean=2.09,std=0.92
kurtosis=0.08,outliers=4.49,max=6.37
skew=0.43
duration
duration
min=-0.47,mean=0.02,std=0.22
kurtosis=0.08,outliers=-1.78,max=1.03
skew=0.43

Section 8: predictive model deployment

Choice of metric:

There are various metrics that can be applied to evaluate the performance of predictive models. Among these metrics are accuracy, precision, recall, f1, log-loss, roc-auc curve and R2. The prevailing logic in picking the metrics that suits the project's problem description and optimize the models accordingly.

In this project, the ultimate goal is to have the marketing team target costumers that are likely to invest in the deposit term product. Since the point is efficient management of the team's resources, we don't want the team to waste time on costumers who wouldn't invest. In the context of the confusion matrix, this means that the main concern is having a lot of False positives. Precision is our metric of choice since it measures the number of true positives divided by number of true and false positives.

Predictive model used → XGBoost classifier

Reason of choice: At the heart of it, it is still a group of decision trees, but what makes it superior in terms of performance is that it uses gradient descent to fix errors of trees that joined the model as it goes and keeps adding trees until no further improvements can be made.

Scaling: Doesn't require scaling similar to all tree-based models

Hypothesis (what can the model handle well): The usage of the hyperparameter scale_pos_weight pays more attention to misclassifications of the minority class. This is specifically helpful in cases of severe imbalance.

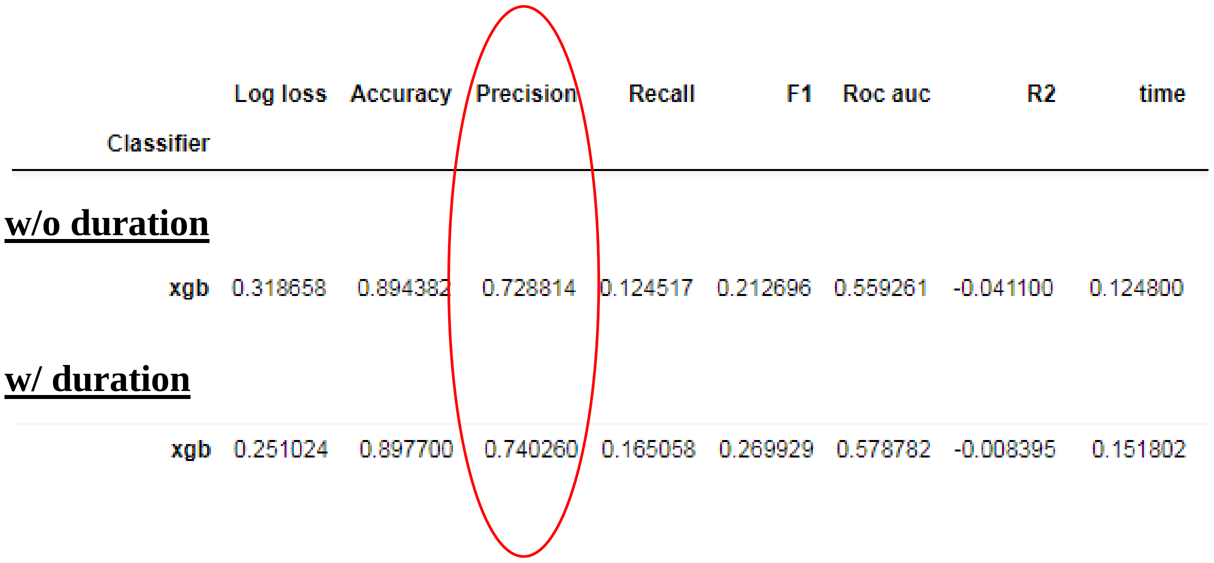
Hyperparameter tuning: we set the scale_pos_weight relative to the ratio between the positive to negative class in the training set (estimate), encouraging the model to over-correct the positive class.

- Estimate=total majority examples/total minority example
  **w.r.t the training dataset only

```
Counter({0: 31913, 1: 4251})

Estimate: 7.507
```

Chosen hyperparameter value: Since our most important metric of performance is precision, it makes sense that the pos_weight that performs best is the one that gives most advantage to the minority class.

$$pos_{weight} = \frac{2}{estimate}$$



Classifier	Log loss	Accuracy	Precision	Recall	F1	Roc auc	R2	time
w/o duration								
xgb	0.318658	0.894382	0.728814	0.124517	0.212696	0.559261	-0.041100	0.124800
w/ duration								
xgb	0.251024	0.897700	0.740260	0.165058	0.269929	0.578782	-0.008395	0.151802

As expected, this choice of weight does improve the detection of the minority class, but it also confuses the majority class quite a lot. We believe this trade-off is still sensible in our problem. This result means that if the marketing team focus on the positively-predicted costumers, 72% of them will go-for the deposit product.

It's important to mention that the previous result is obtained while excluding *duration* feature. This feature is problematic from a logical perspective since the duration of the call is registered After the attempt to convince the costumer is already concluded. So the outcome of the conversation is already known as well. As expected, the precision improves when duration is taken in consideration, though adding it to the analysis is not logistically sound. The improvement in recall is even bigger, this is because there's an implied definition that precision is used to analyze how well a model will predict a future event, while recall is used to evaluate how well a previous event was predicted. Because of the previously discussed thought that duration is known only when contact with a costumer is in the past, this agrees with improving recall.